2Г

УДК 004.272.43

В.А. Гудков
НИИ многопроцессорных вычислительных систем им. акад. А.В. Каляева
Южного федерального университета, г. Таганрог, Россия
gudkov@mvs.tsure.ru

Методы синхронизации потоков данных в структурной составляющей параллельной программы на языке COLAMO

В статье рассматриваются особенности языка высокого уровня COLAMO и возможность его применения для программирования реконфигурируемых вычислительных систем. Приведены методы синхронизации потоков данных в структурной составляющей параллельной программы на языке COLAMO.

Применение реконфигурируемых вычислительных систем (РВС) для решения задач различных классов, таких как линейная алгебра, математическая физика, символьная обработка, зачастую оказывается гораздо более эффективным по сравнению с использованием традиционных многопроцессорных вычислительных систем (МВС) кластерной архитектуры.

Вместе с тем широкое использование РВС сдерживается сложностью их программирования, так как помимо разработки параллельной программы, организующей потоки данных, программисту необходимо задать и конфигурацию вычислительной системы, на которой задача будет решаться с наибольшей эффективностью.

Такой подход к программированию требует от программиста наличия дополнительных навыков в области проектирования вычислительных устройств на языках описания аппаратуры, в частности, на VHDL, что, как правило, приводит к необходимости подключения специалистов-схемотехников к программированию прикладной задачи на РВС. Для описания схемотехнической реализации и организации потоков данных в РВС в едином синтаксисе разработан и используется язык программирования высокого уровня COLAMO [1].

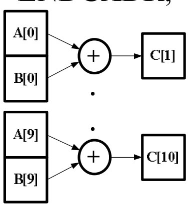
Целью настоящей статьи является рассмотрение языка высокого уровня COLAMO и основных этапов трансляции параллельных программ на языке COLAMO в структурную составляющую, представляющую собой аппаратную реализацию информационного графа, адекватного структуре транслируемой задачи. Особое внимание уделено проблеме синхронизации и стробировании потоков данных в информационном графе структурной составляющей параллельной программы.

Фундаментальным типом вычислительной структуры в языке COLAMO является конструкция «кадр». Кадром является программно-неделимая компонента, представляющая собой совокупность арифметико-логических команд, выполняемых на различных элементарных процессорах, обладающих распределенной памятью и соединенных между собой в соответствии с информационной структурой алгоритма таким образом, что вычисления производятся с максимально возможными параллелизмом и асинхронностью. Кадр фактически определяет вычислительную структуру и потоки данных в РВС в данный момент времени. При этом все операции в теле кадра выполняются асинхронно с максимальным параллелизмом, а последовательность смены кадров однозначно определяется программистом.

В языке отсутствуют явные формы описания параллелизма. Распараллеливание достигается с помощью объявления типов доступа к переменным и индексации элементов массивов. Для исключения конфликтов одновременного чтения и записи ячеек памяти в пределах текущего кадра используется широко распространенное в языках потока данных правило единственной подстановки: переменная, хранящаяся в памяти, может получить значение в кадре только один раз.

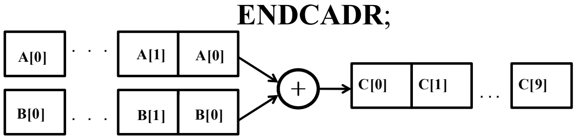
Для обращения к данным используются два основных метода доступа: параллельный доступ (задаваемый типом Vector) и последовательный доступ (задаваемый типом Stream). На рис. 1 представлены программы, являющиеся граничными примерами извлечения параллелизма, и графы синтезируемых вычислительных структур.

```
VAR A,B,C: Integer [10 : Vector] Mem;
VAR I : Number;
    CADR SummaVector;
       For I := 0 to 9 do
         C[I] :=A[I]+B[I];
    ENDCADR;
```



а)

```
VAR A,B,C : Integer [10 : Stream] Mem;
VAR I : Number;
    CADR SummaStream;
       For I := 0 to 9 do
         C[I] :=A[I]+B[I];
    ENDCADR;
```



б)

Рисунок 1 – Параллельное и последовательное сложение массивов

Тип доступа Stream указывает на последовательную обработку элементов одномерного массива, а тип Vector позволяет обрабатывать элементы одномерного массива одновременно.

Многомерные массивы состоят из множества измерений, каждое из которых может иметь последовательный или параллельный тип доступа, задаваемый ключевым словом Stream или Vector соответственно.

Применение неявного описания параллелизма за счет задания типа доступа позволяет достаточно просто управлять степенью распараллеливания программы, на уровне описания структур данных дает возможность программисту максимально просто описывать различные виды параллелизма в достаточно сжатом виде.

Трансляция программы на языке высокого уровня COLAMO состоит в создании схемотехнической конфигурации вычислительной системы (структурной составляющей) и параллельной программы, управляющей потоками данных (потоковой и процедурной составляющих).

Создание структурной составляющей заключается в построении информационного графа, соответствующего описанным на COLAMO информационным зависимостям между результатами вычислений. При этом для каждой используемой в тексте программы операции подставляется специализированный вычислительный блок в зависимости от типа доступа к переменным, типа данных и т.д. Синтезированный информационный граф задачи передается в среду разработки вычислительных структур Fire!Constructor для укладки на множество ПЛИС РВС и обеспечения синхронизации между ПЛИС [2].

Результатом работы среды Fire!Constructor являются файлы VHDL-описаний для корпусов ПЛИС, содержащих синхронизированные между собой фрагменты вы-

числительного графа задачи. Каждый из созданных файлов далее передается в среду проектирования Xilinx ISE, в результате работы которой создаются схемотехнические конфигурации для каждого корпуса.

Во время синтеза информационного графа транслятор COLAMO выполняет оптимизацию созданного графа, в частности, для сокращения числа ярусов арифметического выражения путем преобразования его в ярусно-параллельную форму (ЯПФ).

Преобразование арифметического выражения к ЯПФ ориентировано на обработку произвольных комбинаций арифметических операций и последующего построения арифметического выражения в виде дерева, корнем которого является итоговый результат арифметического выражения.

Алгоритм преобразования арифметического выражения к ЯПФ состоит из следующих основных шагов: обработки вычислительных операций математического выражения и представления математического выражения в виде дерева.

Обработка вычислительных операций математического выражения заключается в:
– определении необходимого количества проходов n по обрабатываемому выражению;
– обработке $к$-й операции, где $к$ – номер арифметической операции от начала выражения.

Количество проходов n определяется как максимальное значение n, при котором выполняется следующее условие: $2^{n-1} <= \left[\frac{CountOperation}{2}\right]$ при условии, что CountOperation > = 2, в противном случае $n = 0$, где CountOperation – количество вычислительных операций (суммирования и вычитания) в математическом выражении.

Правило обработки $к$-й операции.

Обработка выражения начинается со второй операции в выражении (то есть $к = 2$).

В зависимости от значения $к$-й операции от начала выражения будет зависеть значение операции $m = к+2^n$. Если $к$-я операция имеет значение бинарной операции вычитания, а операция с номером $m = к+2^n$ имеет тот же приоритет, что и операция $к$, то значение операции $m = к+2^n$ меняется на противоположное.

Во всех остальных случаях выполняется переход на операцию с номером $m = к+2^{n+1}$.

Построение дерева математического выражения (рис. 2):
– на первой итерации все исходные данные математического выражения разбиваются на пары, и для каждой пары вычисляется их значение;
– далее все полученные значения пар также разбиваются на пары, и снова выполняется вычисление значений пар и т.д.

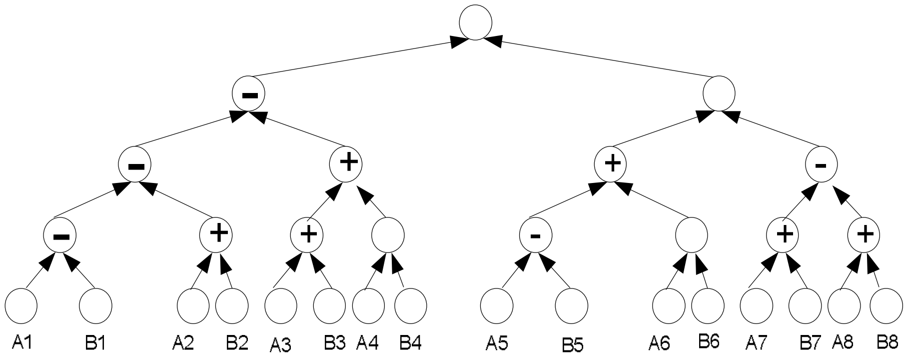


Рисунок 2 – Дерево математического выражения

Построенное дерево представляет собой информационный граф математического выражения, вершинами которого являются операционные вершины (элементарные

процессоры) и информационные вершины (информационные вершины – контроллеры распределенной памяти, регистры и т.д.), а дугами – потоки данных.

После построения информационного графа транслятор выполняет синхронизацию и стробирование потоков данных в графе.

Синхронизация потоков данных в информационном графе осуществляется расстановкой временных задержек.

Элементы временных задержек по функциональному назначению подразделяются на следующие:

а) задержка операционной вершины графа;
б) алгоритмическая задержка (DifferenceIndex);
в) синхронизационная задержка (Koef).

Элемент задержки операционными вершинами графа определяется латентностью данной вершины и применяется для синхронизации потоков данных между операционными вершинами информационного графа. Расстановку элементов задержек, вносимых операционными вершинами графа, выполняет среда Fire!Constructor.

Элемент алгоритмической задержки определяется алгоритмом задачи и возникает при одновременном доступе к различным элементам массива, расположенным в одном канале памяти.

Элемент синхронизационной задержки используется для согласования различных алгоритмических задержек между различными контроллерами распределенной памяти с целью выравнивания потоков данных.

Таким образом, для синхронизации потоков данных в информационном графе на этапе трансляции параллельной программы транслятор выполняет расчет временной задержки для каждого канала памяти путем суммирования алгоритмической и синхронизационной задержек.

Алгоритм расчета временных задержек выглядит следующим образом:

a) Выполнить определение множества MaxMin = $\{V_0, V_1, \dots V_n\}$, где n – количество информационных вершин, а V_i – это кортеж <Variable, DifferenceIndex>, в котором Variable – соответствует информационной вершине информационного графа;

b) Выполнить расчет значения DifferenceIndex для всех элементов множества MaxMin;

c) Определить максимальный кортеж M из множества MaxMin;

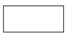
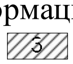
d) Выполнить расчет Koef для всех элементов множества MaxMin относительно кортежа M;

e) Вычислить Delay = DifferenceIndex + Koef для всех элементов множества MaxMin;

f) Выполнить расстановку вычисленных временных задержек для всех дуг информационного графа, принадлежащих картежу V_i из множества MaxMin.

На рис. 3 показан информационный граф программы с расставленными временными задержками.

Здесь квадратные блоки соответствуют входным и выходным информационным вершинам графа.

Вершины графа, обозначенные светлыми кружками, являются арифметико-логическими операциями, реализованными на ЭП, а вершины, обозначенные в виде светлого прямоугольника , указывают на элемент алгоритмической временной задержки информации на определенное число тактов, а в виде заштрихованного прямоугольника  – на элемент синхронизационной временной задержки.

В полученном информационном графе выполняется замена арифметических операций (сложения, умножения, деления и т.д.) на эквивалентные им библиотечные элементы (сумматор, умножитель, делитель и т.д.).

```
VAR A,B,C,D : Integer [100 : Stream] Mem;
VAR I : Number;
CADR SummaStream;
  For I := 5 to 90 do
D[I]:=((A[I]+A[I-5])+B[I])*(C[I]-C[I-2]);
ENDCADR;
```

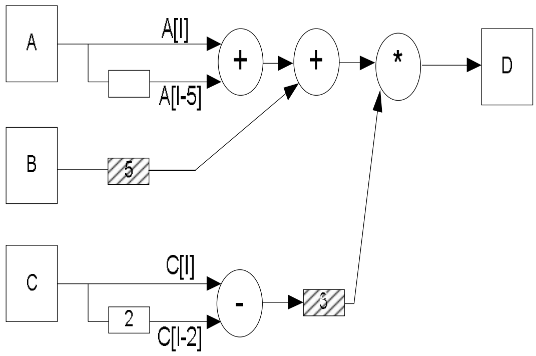


Рисунок 3 – Программа и эквивалентный ей информационный граф с расставленными временными задержками

Для корректной обработки потоков данных, проходящих через информационный граф, представленный в виде библиотечных элементов, необходимо выполнить стробирование потоков данных.

Стробирование данных осуществляется сигналом «Маркер», который подключен к определенным библиотечным элементам в информационном графе и сопровождает поток данных, указывая на их валидность.

Алгоритм подключения «Маркеров» в информационном графе выглядит следующим образом:

– Найти независимые информационные подграфы в информационном графе;
– Определить множество MaxMin согласно пункту а) алгоритма расчета временных задержек, вносимых данными для всех информационных вершин информационного подграфа;
– Вычислить максимальный кортеж M из множества MaxMin;
– Найти информационную вершину в подграфе, соответствующую кортежу M;
– Выполнить перебор всех вершин подграфа и подключить соответствующий библиотечным элементам сигнал «Маркер».

На рис. 4 показан информационный граф, представленный в виде библиотечных элементов с подключенными сигналами «Маркер», где к блоку КРП C на вход MI подключен сигнал «Маркер» блока КРП A с выхода MO.

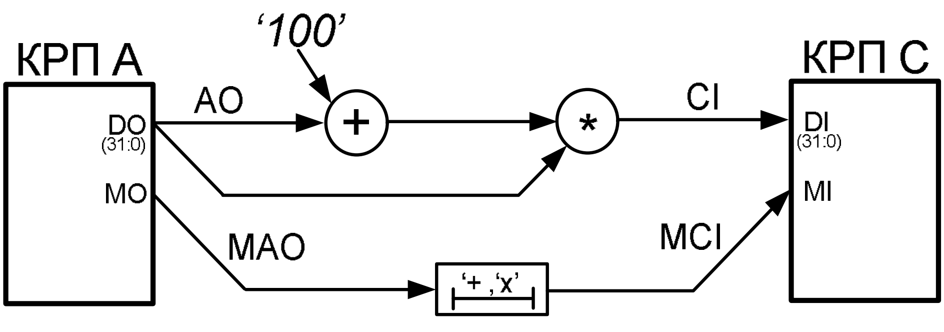


Рисунок 4 – Информационный граф с подключенными сигналами «Маркер»

Полученный информационный граф, представленный в виде специализированных функциональных библиотек, передается в среду разработки вычислительных структур Fire!Constructor, которая выполняет размещение данного графа задачи на архитектуру МВС. Результат представления информационного графа в среде Fire!Constructor представлен на рис. 5.

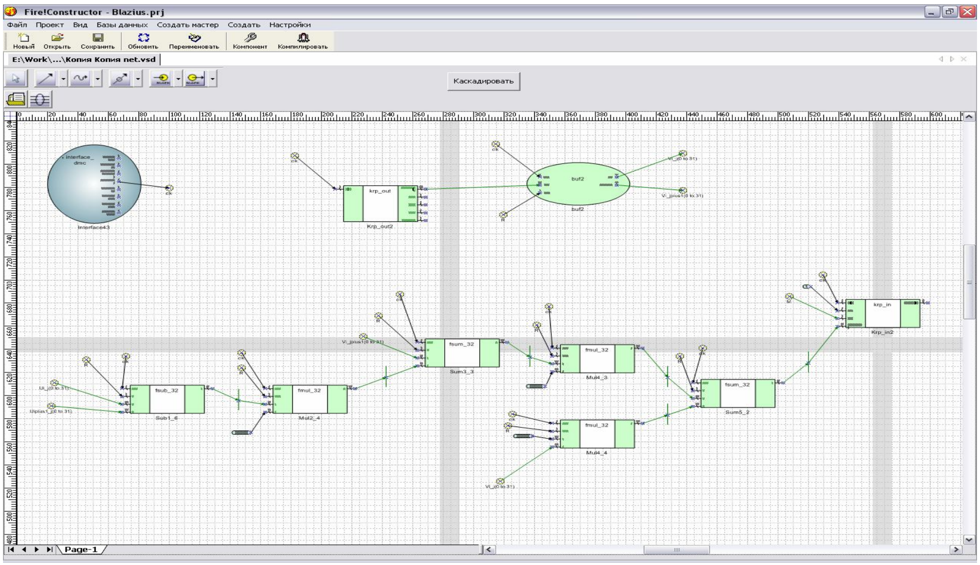


Рисунок 5 – Информационный граф задачи в среде Fire!Constructor

Среда Fire!Constructor выполняет размещения информационного графа задачи на архитектуру РВС и его трансляцию в VHDL-описание, используемое для создания конфигурационных файлов, загружаемых в ПЛИС РВС.

Такой подход программирования реконфигурируемых вычислительных систем освобождает программиста от построения графа задачи в виде функциональных библиотек в среде Fire!Constructor и организации потоков данных в РВС, сократив время создания параллельных программ для РВС в 3 – 10 раз, и исключает участие специалиста-схемотехника при разработке параллельных прикладных программ.

Литература

1. Реконфигурируемые мультиконвейерные вычислительные структуры [Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.] / [под общ. ред. И.А. Каляева]. – Ростов н/Д : Издательство ЮНЦ РАН, 2008. – 320 с.
2. Гуленок А.А. Среда разработки масштабируемых компонентов вычислительных структур и отображения их на архитектуру реконфигурируемых систем / А.А. Гуленок // Материалы Третьей ежегодной науч. конф. студентов и аспирантов базовых кафедр ЮНЦ РАН. – Ростов-на-Дону : Изд-во ЮНЦ РАН, 2007. – С. 136-137.

В.О. Гудков

Методи синхронізації потоків даних у структурній складовій паралельної програми мовою COLAMO

У даній статті розглядаються особливості мови високого рівня COLAMO та можливість її використання для програмування реконфігурірує́мих обчислювальних систем. Наведені методи синхронізації потоків даних у структурній складовій паралельної програми мовою COLAMO.

Статья поступила в редакцию 04.06.2009.